Hydro Electronic Devices, Inc.

Composer™ Presto™ USER'S MANUAL

Preliminary

®

IMPORTANT NOTICE - PLEASE READ FIRST

THIS SOFTWARE IS USED AT YOUR OWN RISK. IMPROPER USE OF THIS SOFTWARE MAY RESULT IN IMPROPER FUNCTIONING OF A MODULE. THE USER IS SOLELY RESPONSIBLE FOR THE RESULTS OF ANY USE OF THE SOFTWARE. HED MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THE SOFTWARE, ITS USE, OR ANY RESULT THAT MAY BE OBTAINED THROUGH ITS USE. USE OF THE SOFTWARE REQUIRES PROFESSIONAL JUDGMENT AND IT IS SOLELY THE USER'S RESPONSIBILITY TO ASSESS THE APPROPRIATENESS OF ANY APPLICATION OF THE SOFTWARE. HED WILL NOT BE LIABLE TO THE USER OR ANY THIRD PARTY FOR ANY DAMAGES OF ANY KIND, DIRECT, CONSEQUENTIAL OR OTHERWISE, REGARDLESS OF THE LEGAL THEORY, ARISING FROM OR ASSOCIATED WITH THE SOFTWARE OR ITS USE.

DISCLAIMER OF WARRANTY.

LIMITATION OF LIABILITY. UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, TORT, CONTRACT, OR OTHERWISE, SHALL VENDOR OR ITS LICENSORS BE LIABLE TO YOU OR ANY OTHER PERSON FOR (AND VENDOR HEREBY EXPRESSLY DISCLAIMS ANY AND ALL LIABILITY FOR) ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT LIMITATION, DAMAGES FOR WORK STOPPAGE, COMPUTER FAILURE OR LOSS OF REVENUES, PROFITS, GOODWILL, USE, DATA OR OTHER INTANGIBLE OR ECONOMIC LOSSES. IN NO EVENT WILL VENDOR OR ITS LICENSORS BE LIABLE FOR ANY DAMAGES IN EXCESS OF THE AMOUNT PAID TO LICENSE THE SOFTWARE, EVEN IF YOU OR ANY OTHER PARTY SHALL HAVE INFORMED VENDOR OR ITS LICENSORS OF THE POSSIBILITY OF SUCH DAMAGES, OR FOR ANY CLAIM. NO CLAIM, REGARDLESS OF FORM, MAY BE MADE OR ACTION BROUGHT BY YOU MORE THAN ONE (1) YEAR AFTER THE BASIS FOR THE CLAIM BECOMES KNOWN TO THE PARTY ASSERTING IT.

INDEMNIFICATION.

FULL TEXT OF THE CANLink® Composer™™ LICENSE IS INCLUDED WITH THE SOFTWARE

Table of Contents

1.0 Introduction

Orchestra's Presto™ enables a user to use all the features Orchestra provides and write their application or part of the application using the C programming language instead of only Orchestra ladder logic. The low level CPU drivers, such as communication protocol, diagnostic tools, I/O management, etc., are all provided as precompiled object code. The user is responsible for setting up a system in Composer™ and using CodeWarrior™ to add user specific code to the C project and compile it. Presto™ is intended for users with C programming experience.

2.0 Material

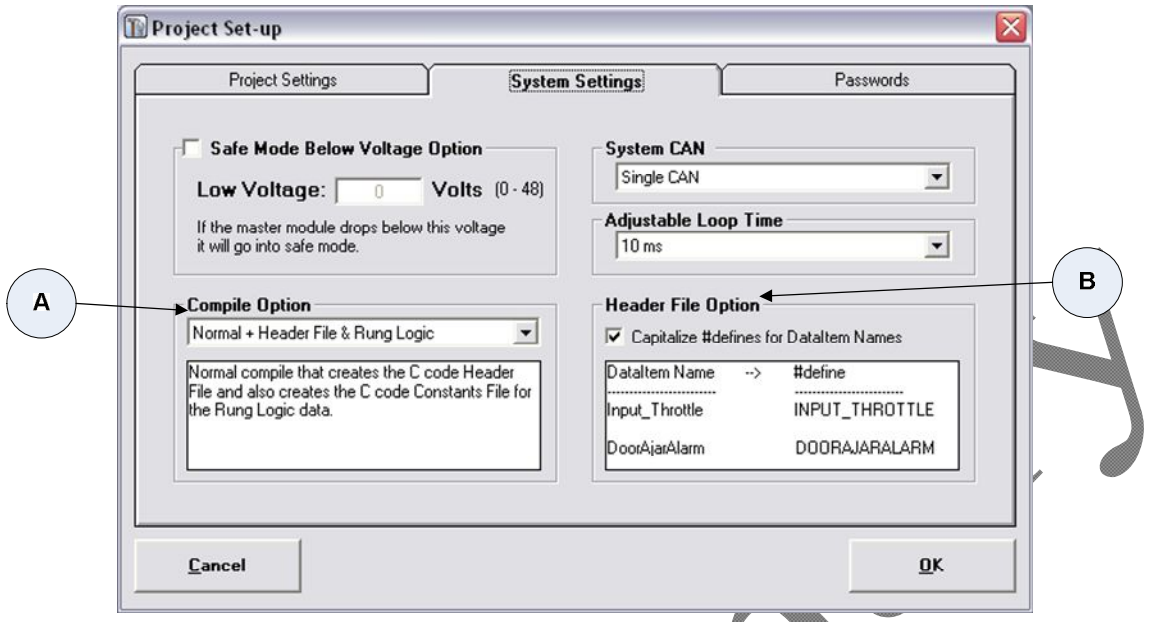
- Required
  - HED® Blue Dongle with Presto™
    - HED® Part Number: CL-012-304
  - Freescale CodeWarrior™ For HCS12(X) Version 4.7 with appropriate license
    - Not included with Orchestra. Needs to be ordered through Freescale
- Optional
  - Wire Harness or Switch Box
    - Contact your supplier for ordering information
    - Create your own
  - CL-802-100
    - Only required if the master module you are using does not have RS232 or USB
    - Can be used even if the master module does have RS232 or USB
  - P&E Multilink
    - Used for debugging software through JTAG
    - If this is not purchased module can still be programmed using Orchestra tools however debugging your software is more difficult
    - Not included with Orchestra. Needs to be ordered through P&E Micro
    - Note: opening the module to plug the Multilink in will void the module's warranty

See Appendix C for connection information

3.0 Setup

Using Composer™ to create header files for a C project is almost identical to a standard Composer™ only system. The only difference is the "Compile Option" has to be changed in the Project Set-up window's System Settings tab.

To get to the Project Setup window press the  icon or under the "Options" menu – click "Edit Project Setup".



A) The options are:

- Normal – Creates only standard Composer™ files.
- Normal + Header File – Creates standard Composer™ files and the header files for the C program. However even if there are rungs defined in Composer™ they will not be included in the header file project
- Normal + Header File & Rung Logic – Creates standard Composer™ files and the header files for the C program including any rungs that are defined.

B) There is a header file option to convert data item names to all capital letters. This feature is for users that have a coding standard that requires macros to be capitalized. For example:

- input1
  - With capitalization enabled
    - #define INPUT1 IOMap[0]
  - Without capitalization enabled
    - #define input1 IOMap[0]
- Output1
  - With capitalization enabled
    - #define OUTPUT1_VALUE IOMap[0]
    - #define OUTPUT1_FLASH IOMap[1]
    - #define OUTPUT1_STATUS IOMap[2]
    - #define OUTPUT1_CURRENT IOMap[3]
  - Without capitalization enabled
    - #define Output1_VALUE IOMap[0]
    - #define Output1_FLASH IOMap[1]
    - #define Output1_STATUS IOMap[2]
    - #define Output1_CURRENT IOMap[3]

4.0 Compiling a Presto™ Project in Composer™

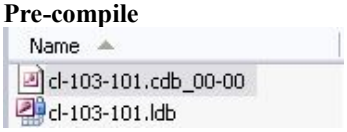
A Presto™ dongle is required to run the Composer™'s Compiler. The type of dongle being used is displayed on the bottom of the Composer™ main window.

The options for Composer™ are

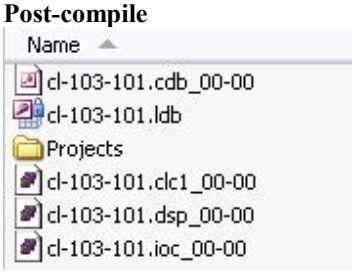
- Level 1 : HF – level 1 dongle with Presto™
- Level 1 – Standard level 1

The Composer™'s Compiler creates two files that are included in your C project, "Constants.h" and "Constants.c". The "Constants.h" file contains the #defines where all the data items are in the IOMap. The "Constants.c" file contains 3 constant arrays that the firmware will use to set up all the modules in the system and run the rungs if activated. The Composer™'s Compiler will also copy the CodeWarrior™ project for the selected master module to the same location as the Composer™ project file. See below:

Pre-compile



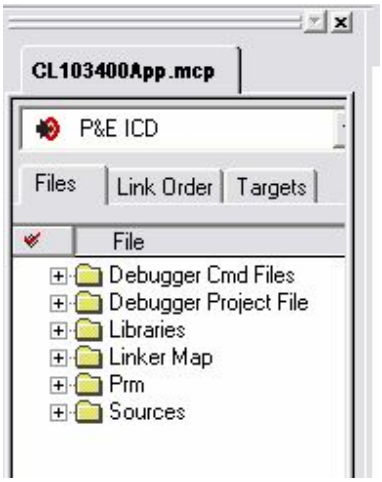
Post-compile



Inside the Projects directory will be a copy of the default-project for the selected master module and the Constants.c and Constants.h files that are specific to the Composer™ project.

5.0 The C project

HED® has provided a starter project that is specific to each master module. Each master module has its own project which contains the files and folders that are displayed in CodeWarrior™. The project view in CodeWarrior™ displays the files and folders as follows:



- Debugger Cmd Files
  - These files are used to setup the P&E debugger
  - These files should not need to be edited however there is information available from CodeWarrior™ should they need to be edited
- Debugger Project Files
  - These files are used to setup the P&E debugger
  - These files should not need to be edited however there is information available from CodeWarrior™ should they need to be edited
- Libraries
  - These files are for the processor
  - These files should not be edited.
- Linker Map
  - Contains the memory map after the project has successfully linked
- Prm
  - Burner.bbl
    - Command for how to build the s19 and phy files
    - This file should not be edited.
  - P&E_ICD_linker.prm
    - This is the linker file for the project. It setups up where all the memory is and where all the different sections of the project go
    - If the processor uses paged memory, 1 page of RAM is reserved for the User application. Therefore this file should not be edited.
- Sources Folder
  - Precompiled Folder
    - Precompiled object files for all the low level firmware
      - Do not edit these files.
  - Constants_.h and .c files
    - The Composer™ Compiler created these files to match the system that was setup in Composer™.
    - Do not edit these files
  - User_Init.h
    - Function Prototype for User_Init().
    - Add user software when applicable
  - User_Init.c
    - Function is called once on startup (See Appendix A for flow chart)
    - Add user software when applicable
  - User_App.h
    - Function Prototype for User_App().
    - Add user software when applicable
  - User_App.c

        - Function is called once per loop (See Appendix A for flow chart)
        - Add user software when applicable
    - User_Can_Receive.h
        - Function Prototype for User_Can_Receive().
        - Add user software when applicable
    - User_Can_Receive.c
        - Function is called once for every CAN message that did not come from a module on the system (See Appendix A for flow chart)
        - Add user software when applicable.
    - User_J1708_Receive.h
        - Function Prototype for User_J1708_Receive().
        - Add user software when applicable
        - Note: This file will only be available on modules that have J1708
    - User_J1708_Receive.c
        - Function is called once for every J1708 message received (See Appendix A for flow chart)
        - Add user software when applicable
        - Note: This file will only be available on modules that have J1708
    - User_Serial_Receive.h
        - Function Prototype for User_Serial_Receive().
        - Add user software when applicable.
        - Note: This file will only be available on modules that have RS232 or USB.
    - User_Serial_Receive.c
        - Funcition is called once for every byte received on any of the SCI lines on the module if the HED® packet are disabled. (See Appendix A for flow chart)
        - Add user software when applicable.
        - Note: This file will only be available on modules that have RS232 or USB.
    - User_Serial_Packet_Receive.h
        - Function Prototype for User_Serial_Packet_Receive().
        - Add user software when applicable.
        - Note: This file will only be available on modules that have RS232 or USB on them.
    - User_Serial_Packet_Receive.c
        - Funcition is called once for every packet received on any of the SCI lines on the module if the HED® packet is enabled. (See Appendix A for flow chart)
        - Add user software when applicable.
        - Packet has to be in the following format [*….] where … is the ACII characters with the user data.
        - Note: This file will only be available on modules that have RS232 or USB.
    - Add additional source and header files as necessary

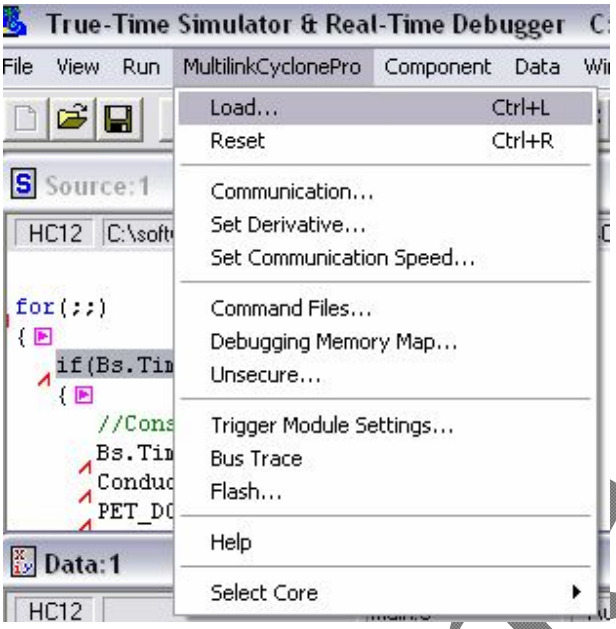
6.0 Compiling and Debugging a C Project

The C project can be compiled and debugged just like a normal CodeWarrior™ project. The CodeWarrior™ documentation has a very thorough explanation of this. Keep in mind that when using the debugger it will erase all the FLASH including the boot code so the HED® downloader will not work correctly. See section 7.0 on reloading the boot code to reprogram the boot code.

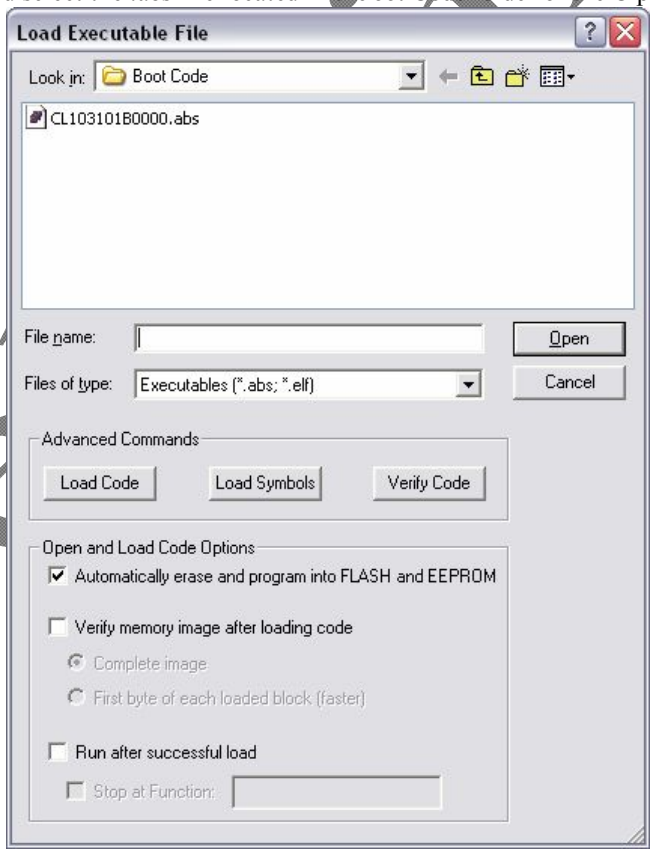
7.0 Reloading the Boot Code

Follow these steps to reload the boot code if it has been erased

1. Open the C project in CodeWarrior™
2. Run the debugger
3. Once it is done programming the module, on the file menu open the Multilink Cyclone Pro > Load



4. Select the open button and select the .abs file located in the Boot Code folder of the C project

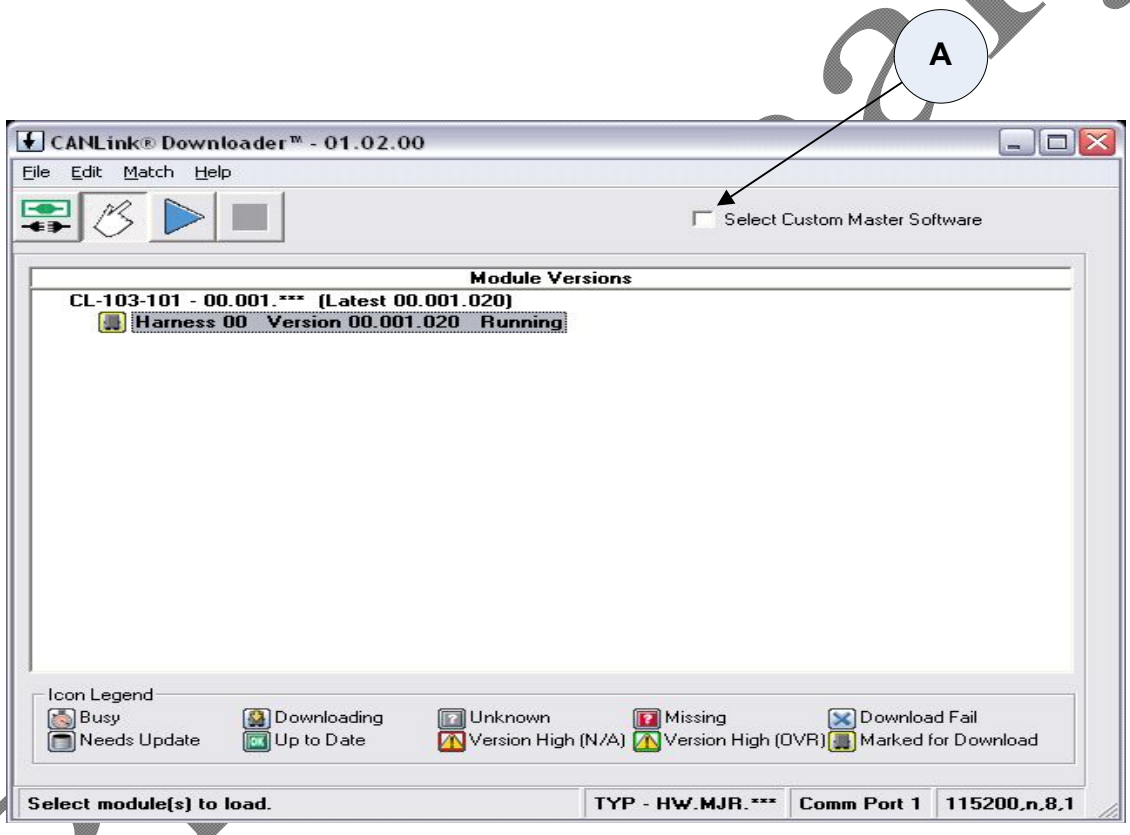


5. It will load the boot code

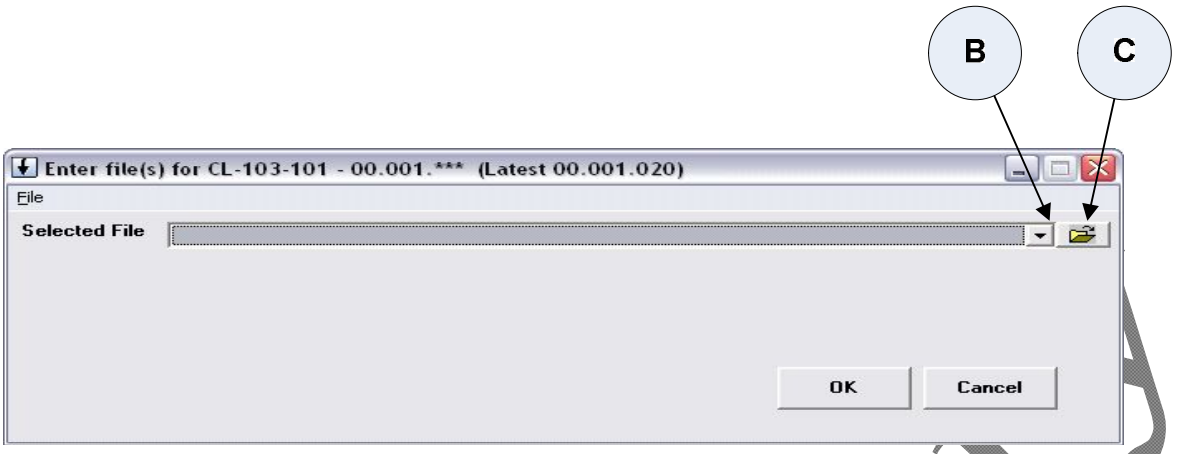
8.0 Reprogramming a unit

A different procedure needs to be followed to reprogram a unit using the Firmware Downloader depending on what software is currently in the master module. There are 3 different options of what it may contain

- Boot Code only
  - Connect with the Firmware Downloader
    - It will go into legacy mode
  - Select the software to download
  - Enter FFFF for the module type
  - Select Start
- Boot Code + Standard Composer™ Software
  - Connect with the Firmware Downloader
  - It will display the master module and all the modules on the bus
  - Click on the master module to select it to be programmed

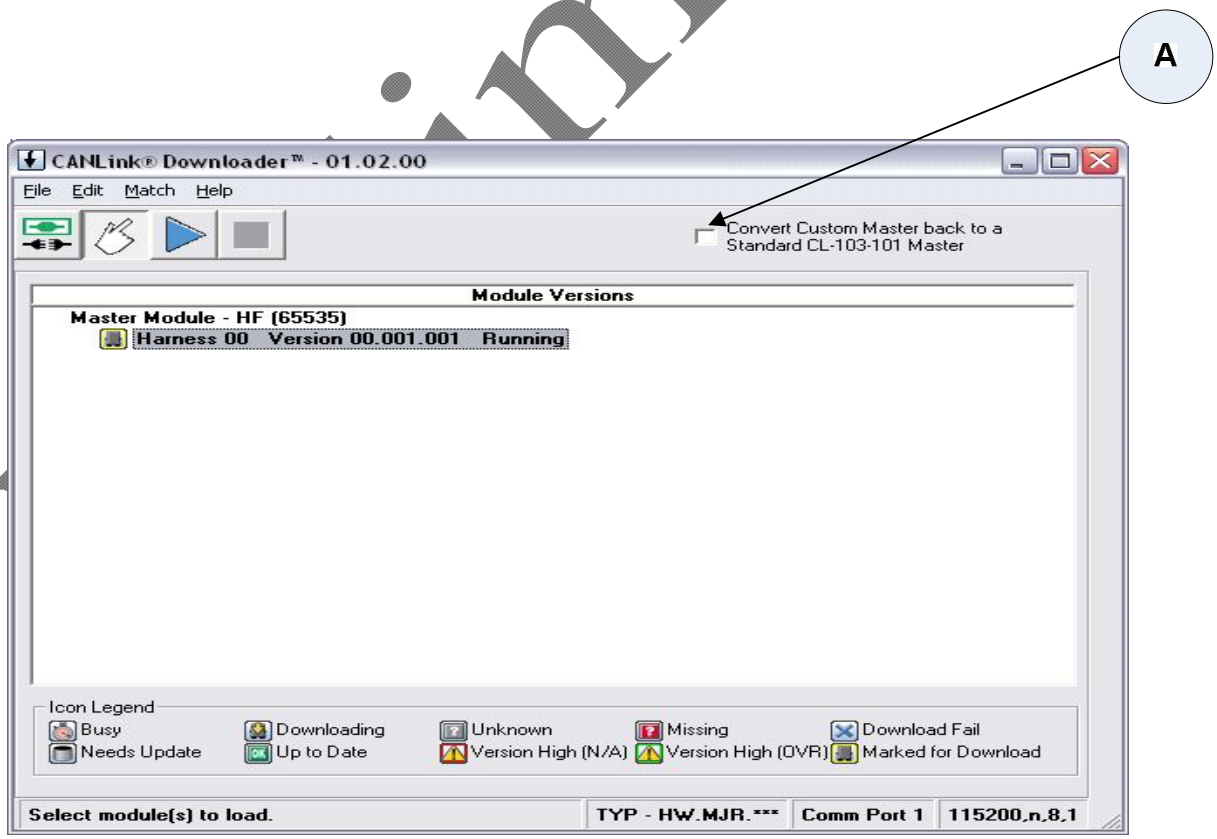


- Check the check box (A) displayed in the upper right to allow selection of custom master software
- Press the start button arrow.
- Select the software to download by:



- Selecting the software from the pull down list(B) of previously selected software
- Pressing the open button(C) and selecting it from a directory
  - Once the software is selected press OK.
  - Wait for the download to finish

- Boot Code + User Software (Created with the C header file)
  - Connect with theFirmware Downloader.
  - It will display the master module as “MasteModule - HF” and allother modules on the bus.
  - Click on the master module toselect it to be programmed.
  - Select Start
  - When prompted select the software to download
  - Select Start



Note: To make the module a standard Composer™ Master click the "Convert Custom Master Back to Standard CL-XXX-XXX Master" checkbox (A).

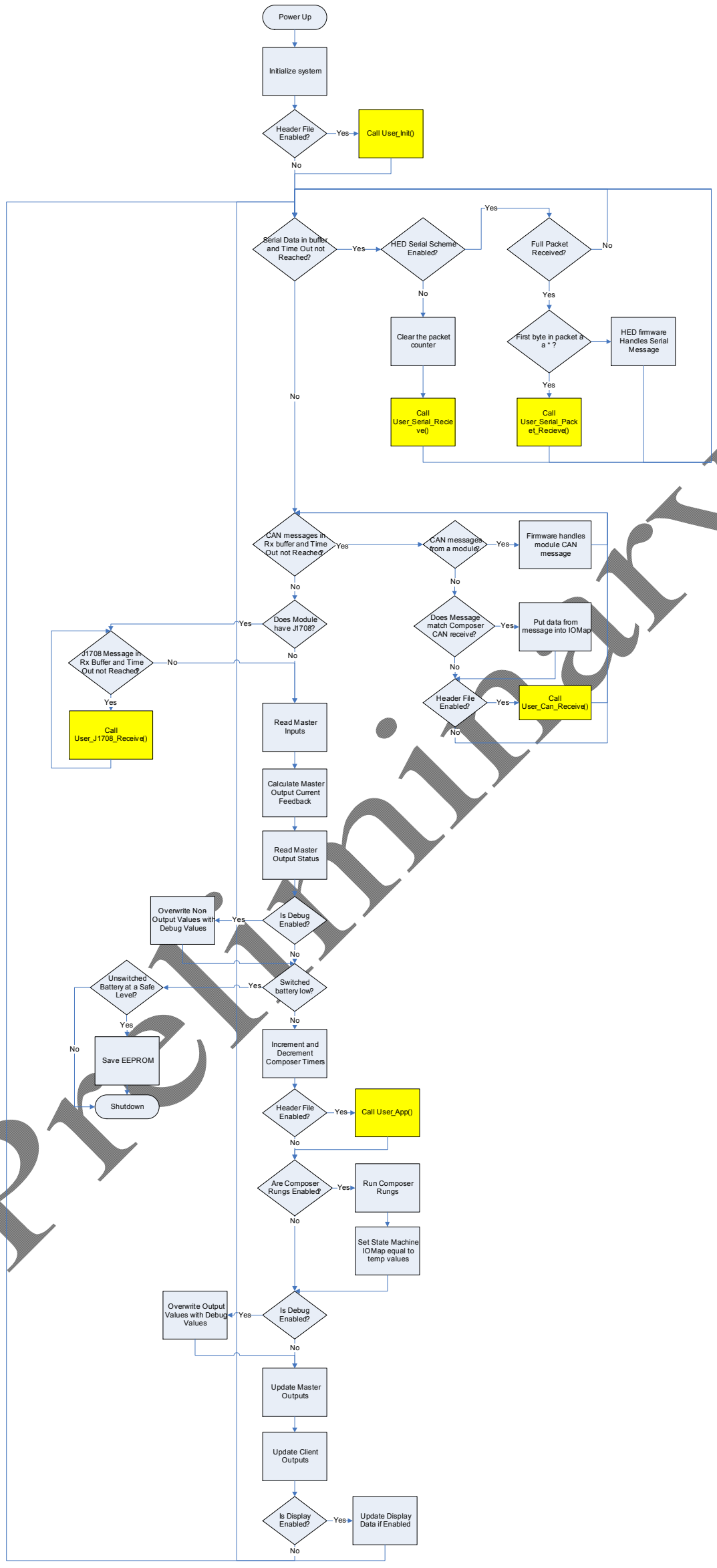
For more information on how to use the Downloader please see the Downloader manual.

9.0 Software notes

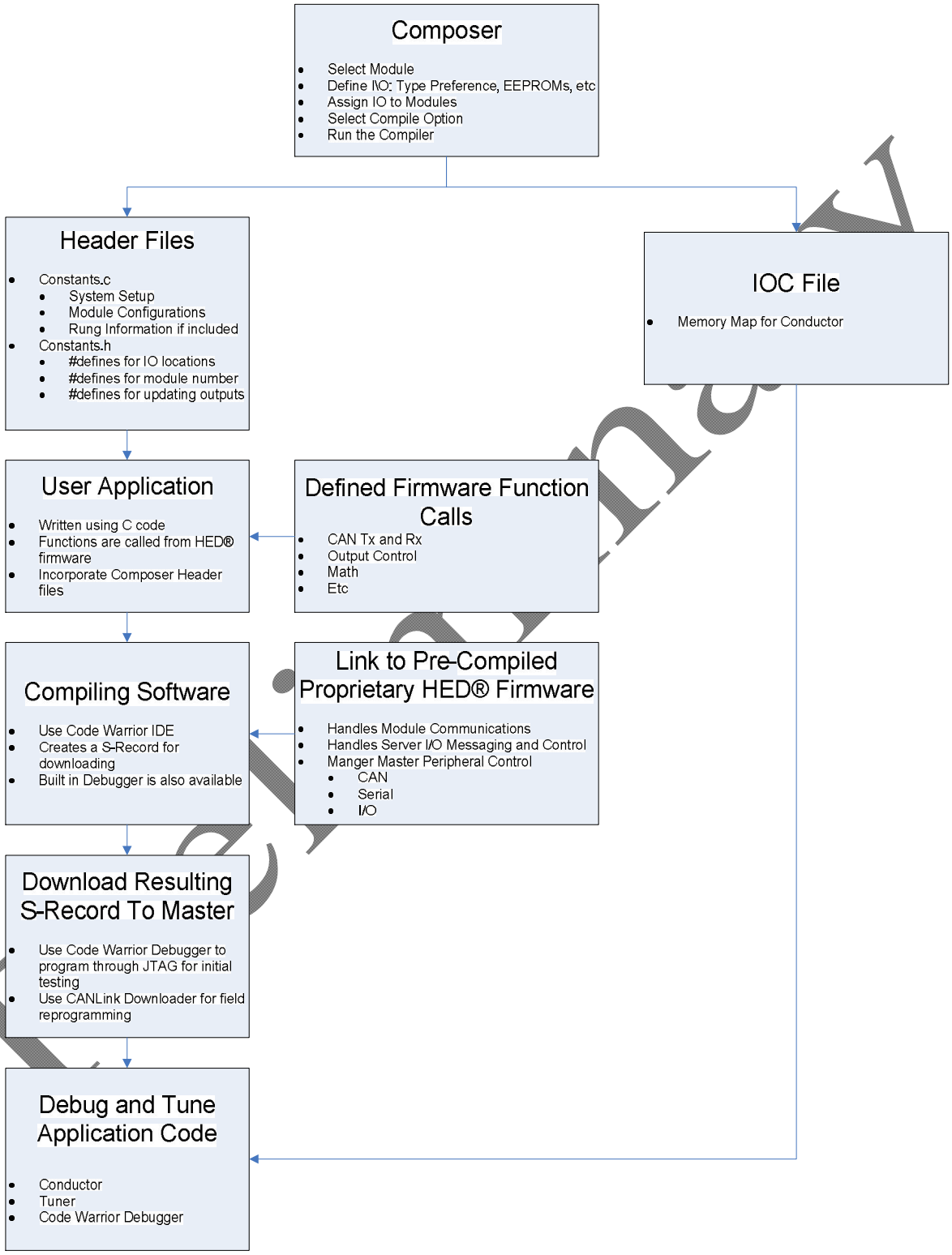
Below are some things to consider when writing the application software

- Flash
  - To reserve a section of flash for data logging, remove it from the linker file. This will ensure that the compiler will not place any of the application code there.
    - Comment out the pages to be used in the paged flash section where it lists all the pages and their range
    - Remove or comment out the pages in the DEFAULT_ROM Section.
  - Use caution when erasing or writing to addresses
    - Using an address that does not exist will cause the processor to reset
    - Erasing an address with application code will cause it too long function properly
- Serial
  - All HED PC programs connections start at 9600 baud rate. If the application changes the baud rate to something other then 9600 ensure that it restores it to 9600 to allow HED tools to connect.
  - Changing the baud rate when a HED tool is connected will cause the tool to timeout and disconnect.
  - HED Serial Packet scheme
    - Start Byte = [
    - End Byte =]
    - User Packet command = *
    - Example
      - [*123]
  - printf function
    - The printf function needs to have the serial line, to send on, defined prior to be used in the application code.
    - Set "SciLineForPrintf" to the desired line.

Appendix A

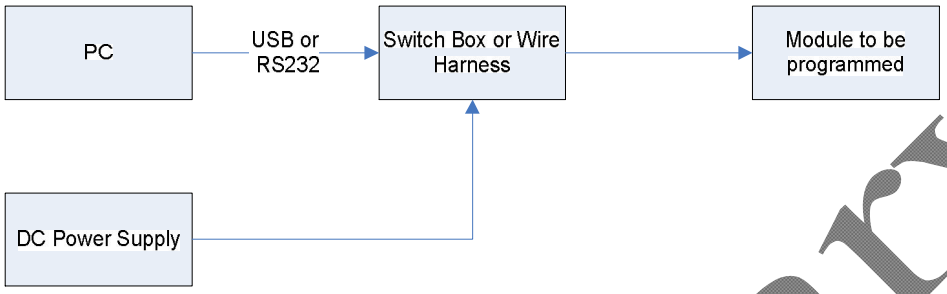


Appendix B



Appendix C

Connection when master module does not have RS232 or USB or if it is preferred to program through CAN.



Connection when master module has RS232 or USB.